EE4302 Advanced Control System CA3

A0260014Y
Guo Shiping

1. In the work in Section 4, the steady-state specification was met by the use of a scaling gain.

Briefly explain the disadvantage, if any, of that approach.

Difficulty in Tuning: In practice, determining the appropriate scaling gain value can be challenging. It often requires extensive tuning and experimentation to find the right gain, and even then, it may not work well under all conditions.
Lack of Adaptability: A scaling gain doesn't adapt to changes in the system or environmental conditions. In dynamic systems with varying disturbances or changes in the plant dynamics, the fixed scaling gain may not provide the desired level of control.

2. New state-variable

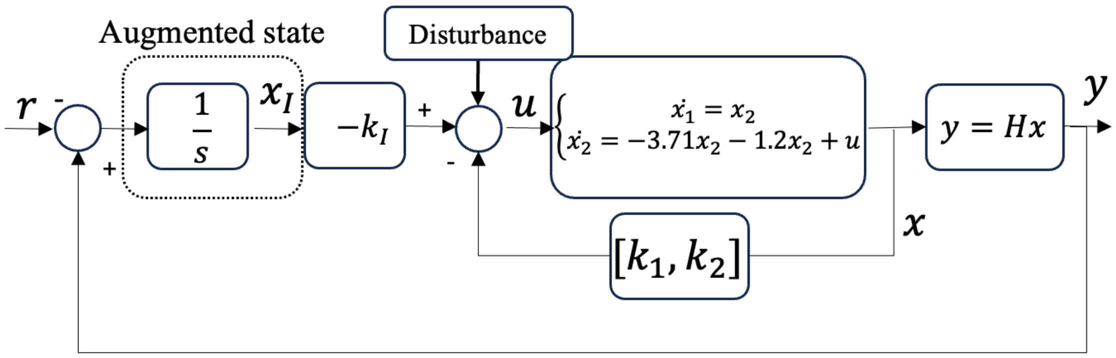

Figure 1 Augmented state structure

Let

$$\dot{x}_I = y - r = x_1 - r$$

New augmented state space form is:

$$\begin{bmatrix} \dot{x}_1 \\ x_2 \\ x_I \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1.2 & -3.71 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_I \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} r + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} v$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_I \end{bmatrix}$$

$$F = A = \begin{bmatrix} 0 & 1 & 0 \\ -1.2 & -3.71 & 0 \\ 1 & 0 & 0 \end{bmatrix}; \ H = C = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$G_u = B_u = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \ G_r = B_r = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}; \ G_v = B_v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

New requirement:

Closed-loop bandwidth:	Not lower than 1.5 rad/s;
Resonant Peak, M_r:	Not larger than 2dB;
Steady-state gain between r and y:	0dB.

Explain carefully why state augmentation will result in a closed-loop system meeting the specs for the two items of steady- state requirement.

Like my setting, the augmenting part is the target, we set

$$\dot{x}_I = y - r = x_1 - r$$

Expanding the state vector introduces extra degrees of freedom in crafting the control law. These additional states facilitate a more precise positioning of the closed-loop poles, a crucial factor in attaining the desired transient and steady-state response characteristics.

When the system achieves stability and converges to zero, the output closely tracks the reference signal. Examining the system structure, the integral action accumulates all the errors.

Why we don't need scaling gain when using augmentation method?

The augmented states include integral action (states representing the integral of the system error), this can inherently address steady-state errors without the need for an additional scaling gain. Integral action ensures that the system continually adjusts to eliminate any steady-state discrepancies.

3. ITAE and Ackerman formula approach

Check the ITAE table and place the desired poles to

1.5*[-0.7081; -0.5210+1.068*1i; -0.5210-1.068*1i]

The next is applying Ackerman formula. The procedure is the same as CA1.

$$\texttt{ans} = s^3 + 2.625\,s^2 + 4.837\,s + 3.374$$

$$W_c = \begin{bmatrix} G & FG & F^2G \end{bmatrix}$$

The controllability matrix is:

```
        0    1.0000   -3.7100
   1.0000   -3.7100   12.5641
        0         0   -1.2000
```

Rank(Wc) = 3. So it's controllable.

$$\texttt{ans} = s^3 + 2.625\,s^2 + 4.837\,s + 3.374$$

```
Sigma = [G F*G F*F*G];
% s^3 + 1.4811s^2 + (1.8193-0.2872i)s+ 0.9006 + 0.2033i
Alpha_F = F^3 + 2.625*F^2 + 4.837*F + 3.374*eye(3)
```

```
Alpha_F = 3x3
     4.6760    7.6623         0
    -9.1948  -23.7513         0
    -4.3644    1.3020    3.3740
```

```
K = [0 0 1]*inv(Sigma)*Alpha_F
```

```
K = 1x3
     3.6370   -1.0850   -2.8117
```

Double check the result:

```
% check ackerman formula result
K_check = acker(F, G, poles)
```

```
K_check = 1x3
     3.6373   -1.0848   -2.8122
```

We can see the result are almost the same.
The simulation results are

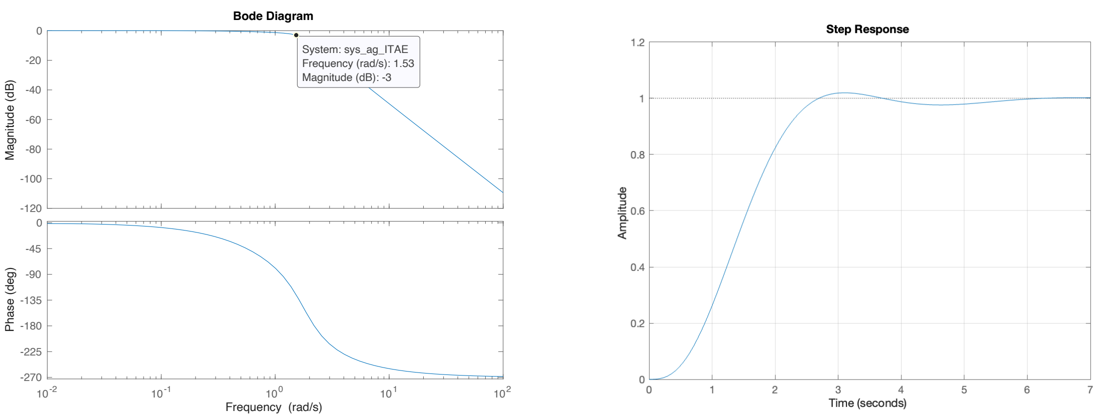

Figure 2 ITAE and Ackerman formula simulation results

4. LQR approach

Keep R consistent as 1, change Q, the diagonal three parameters are R_1, R_2, R_3. Change $R_{1,2,3}$ and watch the changes of frequency response I found that.
Increase R_1, R_2can make the frequency bandwidth decrease.
Increase R_3can make the frequency bandwidth increase.
After many tires I find the following parameters work:

```
Q_LQR = [5,   0,   0;
              0,   1,   0;
             0,   0, 100];
R_LQR = 1;
```

The simulation result

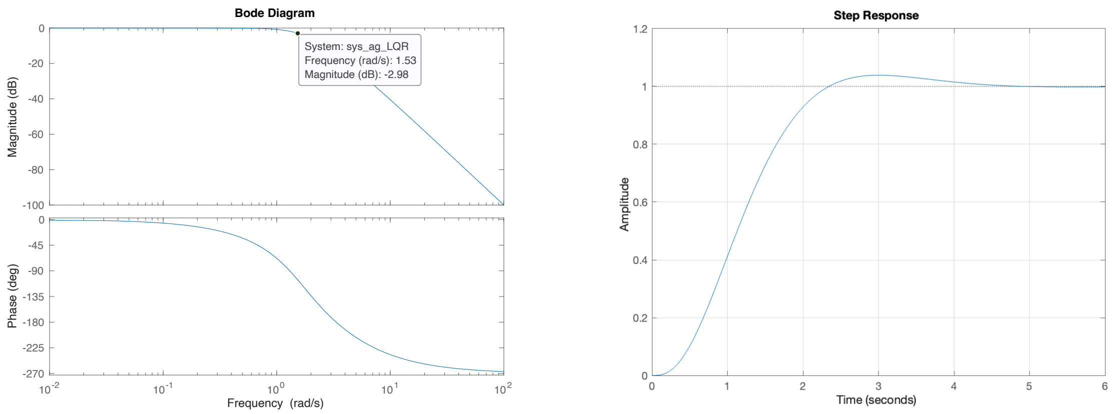

Figure 3 LQR approach simulation results

```
Q_LQR = [5,  0,  0;
        0,  1,  0;
        0,  0, 100];
R_LQR = 1;
[K_LQR,~,~] = lqr(Fbar, Gu, Q_LQR, R_LQR);
sys_ag_LQR = ss(Fbar-Gu * K_LQR, Gr, Hbar, 0);

[num, den] = ss2tf(Fbar-Gu * K_LQR, Gr, Hbar, 0);
sys_tf = tf(num,den)

figure(18)
bode(sys_ag_LQR)
figure(19)
step(sys_ag_LQR)
grid on

sys_ag_LQR_Control = ss(Fbar-Gu*K_LQR,Gr,-K_LQR,0);
% In sys_ag_LQR_Control, Hbar is changed to -K_LQR
% In this case, MATLAB would treat -K_LQR*x as the output of the system,
% which is nothing else but the control signal
% Therefore, by doing a unit step response analysis, the control signal
% response can be displayed
figure(20)
step(sys_ag_LQR_Control)
grid on
```

5. Second-Order Dominant (SOD) approach

We need calculate the desired poles first.
The Resonant Peak calculation formula is

$$M_r = \frac{1}{\sqrt{(2\delta^2)^2 + (2\delta\sqrt{1-2\delta^2})^2}}$$

Let

$$\frac{1}{2\,\sigma\,\sqrt{1-\sigma^2}} < 2$$

The solution is

$$0.26 < \sigma < 0.97$$

Let

$$\sigma = 0.5$$

The bandwidth is

$$w_b = w_n\sqrt{1 - 2\sigma^2 + \sqrt{2 - 4\sigma^2 + 4\sigma^4}} > 2$$

Calculated

$$w_n = 1.6$$

is choose solution.
Then calculate the poles of

$$s^2 + 2\sigma w_n s + w_n^2 = 0$$

Get the poles:

$$\begin{pmatrix} -0.8 - 1.4\,i \\ -0.8 + 1.4\,i \end{pmatrix}$$

The third pole can be chosen far away from these two poles. I choose -5.
So, the three poles are:

$$[-5,\ -0.8 - 1.4i,\ -0.8 + 1.4i]$$

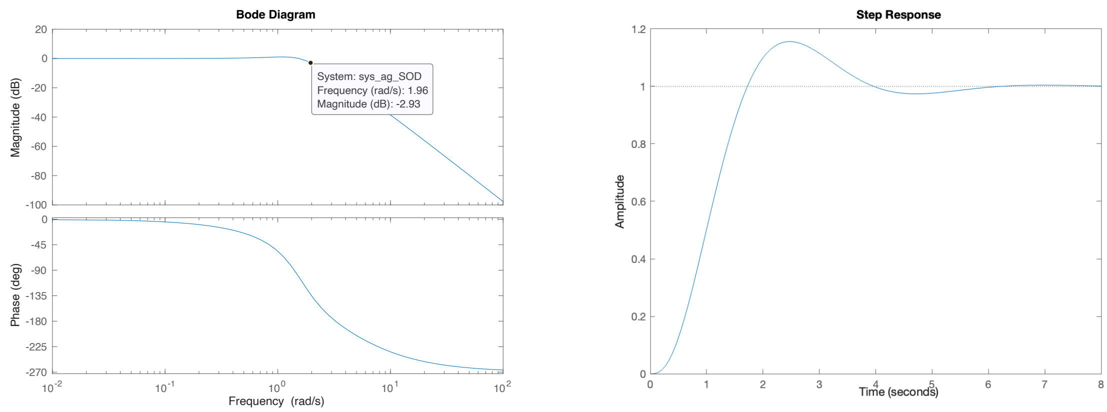

Figure 4 LQR approach simulation results

```
SOD_poles = [-5, -0.8-1.4i, -0.8+1.4i];
F = [0, 1 0;
   -1.2, -3.71 0;
   1 0 0]; % Process Matrix
Gu = [0; 1; 0]; % The input matrix for u
Gr = [0; 0; -1]; % The input matrix for r
Gv = [0; 1; 0]; % Disturbance input
H = [1, 0, 0]; % Output Matrix
K_SOD = acker(F, Gu, SOD_poles);
sys_ag_SOD = ss(F - Gu*K_SOD, Gr, H, 0);
bode(sys_ag_SOD)
```

6. Bessel prototype table methodology

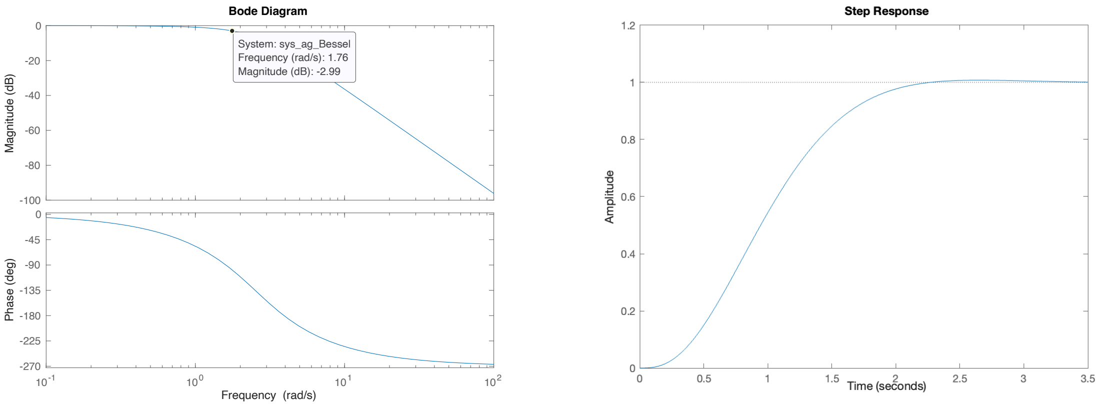

Figure 5 Bessel prototype simulation results

```
BesselPoles = 1.5*[-0.9420;-0.7455+0.7112*1i;-0.7455-0.7112*1i];
K_Bessel = acker(Fbar,Gu,BesselPoles);
sys_ag_Bessel = ss(Fbar-Gu*K_Bessel,Gr,Hbar,0);
figure(14)
bode(sys_ag_Bessel)
figure(15)
step(sys_ag_Bessel)
grid on
```

7. Second-Order Dominant Response methodology.

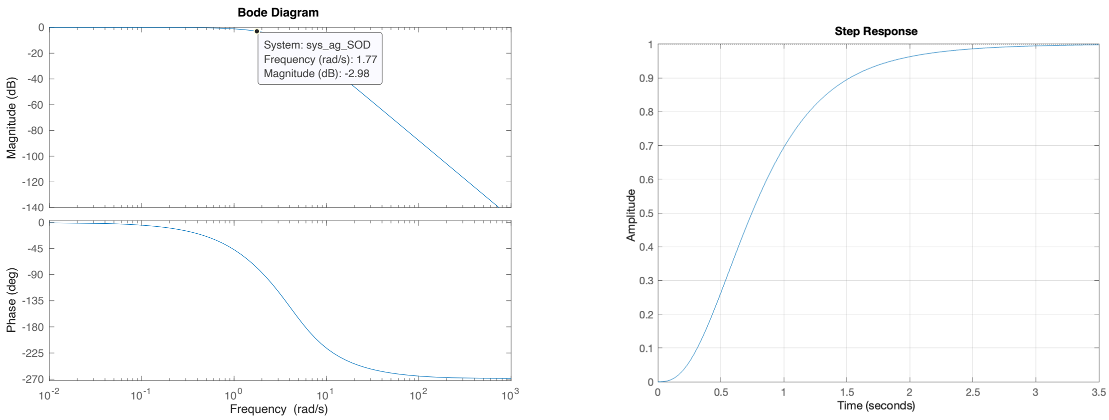

Figure 6 Second-Order Dominant simulation results

```
SODPoles = [-2; -3.6+2.7*1i; -3.6-2.7*1i];
K_SOD = acker(Fbar,Gu,SODPoles);
sys_ag_SOD = ss(Fbar-Gu*K_SOD,Gr,Hbar,0);
figure(16)
bode(sys_ag_SOD)
figure(17)
step(sys_ag_SOD)
grid on
```

8. Disturbance analysis

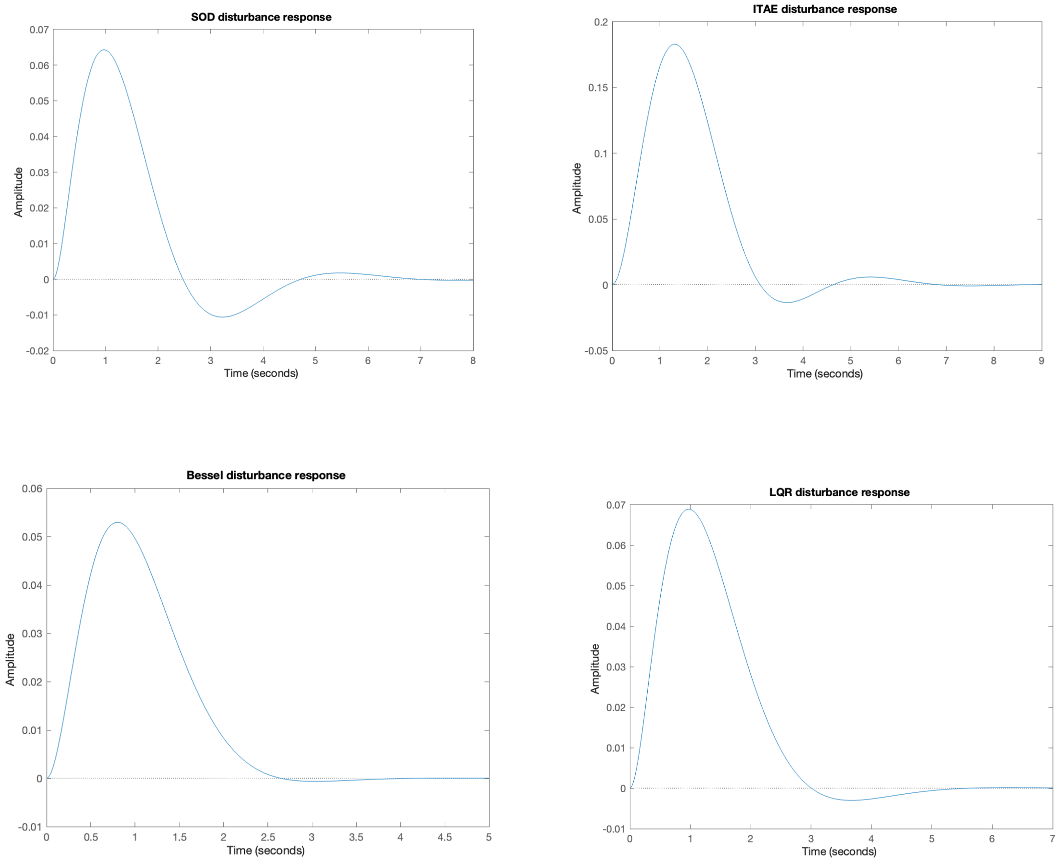

Figure 7 Disturbance comparation and analysis

Analysing the simulation outcomes reveals variations in amplitude and settling time, just for the output response, the ITAE and Bessel approaches exhibiting superior performance compared to other methods. The selection of poles significantly influences the results. Despite methodological differences, all four approaches prove effective, as evidenced by the prompt convergence of the disturbance response to zero.

9. Control signal analysis

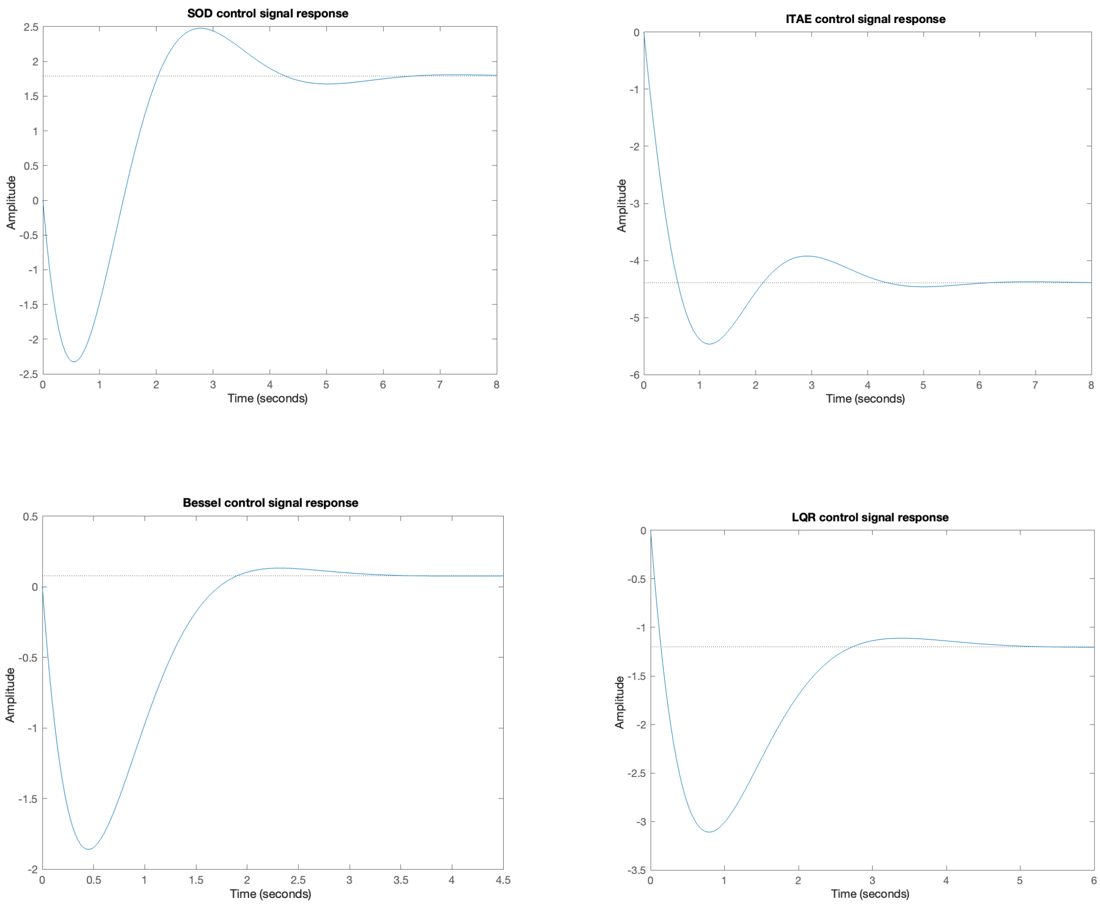

Figure 8 Control signal comparation and analysis

```
%% Response to Control signal
% SOD Output response to control signal
sys_ag_SOD_ctr = ss(F -Gu * K_SOD, Gr, K_LQR, 0);
% Bode plot and step plot without scaling gain
step(sys_ag_SOD_ctr)
title('SOD control signal response')
% ITAE Output response to control signal
sys_ag_ITAE_ctr = ss(F -Gu * K_ITAE, Gr, K_LQR, 0);
% Bode plot and step plot without scaling gain
step(sys_ag_ITAE_ctr)
title('ITAE control signal response')
% Bessel Output response to control signal
sys_ag_Bessel_ctr = ss(F -Gu * K_Bessel, Gr, K_LQR, 0);
% Bode plot and step plot without scaling gain
step(sys_ag_Bessel_ctr)
title('Bessel control signal response')

% LQR Output response to control signal
sys_ag_LQR_ctr = ss(F -Gu * K_LQR, Gr, K_LQR, 0);
% Bode plot and step plot without scaling gain
step(sys_ag_LQR_ctr)
title('LQR control signal response')
```

10. Transfer function comparation

```
sys_tf_SOD =

             13
  ----------------------------
  s^3 + 6.6 s^2 + 10.6 s + 13

sys_tf_ITAE =

               3.375
  ---------------------------------
  s^3 + 2.625 s^2 + 4.837 s + 3.375

sys_tf_Bessel =

               15.63
  --------------------------------
  s^3 + 6.082 s^2 + 15.41 s + 15.63

sys_tf_LQR =

               10
  -------------------------------
  s^3 + 5.886 s^2 + 11.14 s + 10
```

```
[num, den] = ss2tf(F-Gu * K_SOD, Gr, H, 0);
sys_tf_SOD = tf(num,den)
[num, den] = ss2tf(F-Gu * K_ITAE, Gr, H, 0);
sys_tf_ITAE = tf(num,den)
[num, den] = ss2tf(F-Gu * K_Bessel, Gr, H, 0);
sys_tf_Bessel = tf(num,den)
[num, den] = ss2tf(F-Gu * K_LQR, Gr, H, 0);
sys_tf_LQR = tf(num,den)
```

It's interesting to see that all the four transfer functions are similar with each other but the control results are different. It also demonstrates that the poles placement play a very important role to control the system to desired station. Overall, the Bessel method seems better, but we also need consider the control signal and other factors to judge them. Anyway, it's really interesting to see that the small changes of transfer function can bring so much difference.

11. Why does it make sense (assuming that we are operating using an identical "hardware set" in real-life) to require a lower closed-loop bandwidth (1.5 rad/s) here? Experiment, explore and discuss.

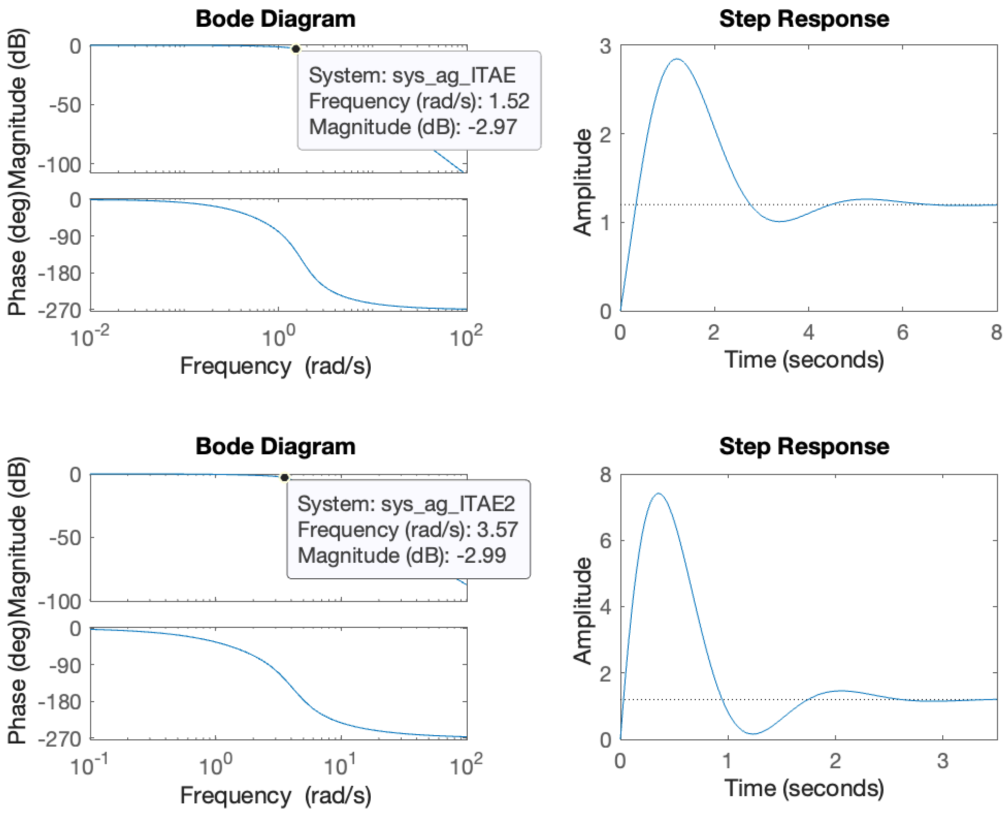

Figure 9 Higher and lower bandwidth comparation

Examining the simulation results, it becomes evident that a higher bandwidth is associated with increased overshoot and greater control energy. In contrast, systems with narrower bandwidths necessitate a lower closed-loop bandwidth. This choice holds relevance in specific scenarios, particularly when working with an identical hardware setup in practical applications. Such decisions often involve trade-offs between performance and stability.

Opting for a lower closed-loop bandwidth can enhance the system's capacity to reject noise and disturbances. This adjustment contributes to improved robustness in the control system by reducing sensitivity to variations in plant dynamics or uncertainties in parameters. Ultimately, lowering the bandwidth can lead to a more stable closed-loop system.

12. Additional exploration

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -3.51x_2 - 0.77x_1 + 1.10u$$
$$y = x_1$$

Explore, discuss, analyze with all suitable simulations, analysis and descriptions in this type of situation on the use of the methods of:

- Using a "Scaling Gain"
- Using the augmented state variable $\dot{x}_I = y - r$

Provide a suitably comprehensive exploration / discussion.

Open loop Bode diagram:

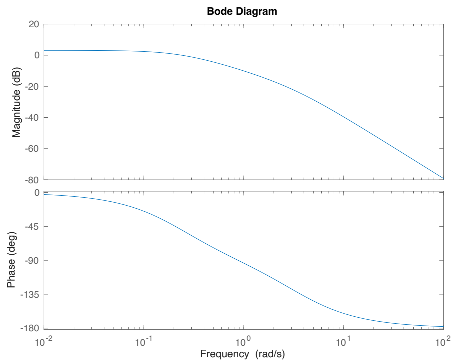

Figure 10 Addition system open loop bode diagram

Suppose set the target bandwidth is bigger than 3dB
Scaling gain approach:

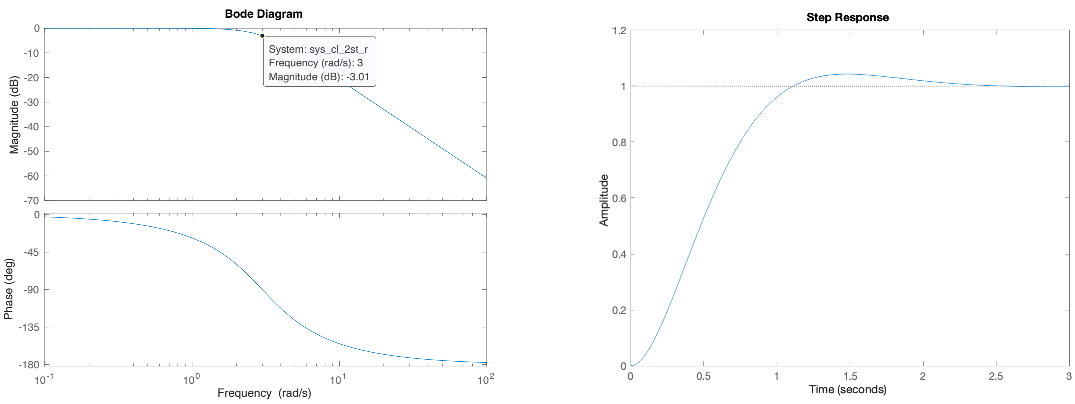

Figure 11 Additional system scaling gain control results

Augment state variable approach:

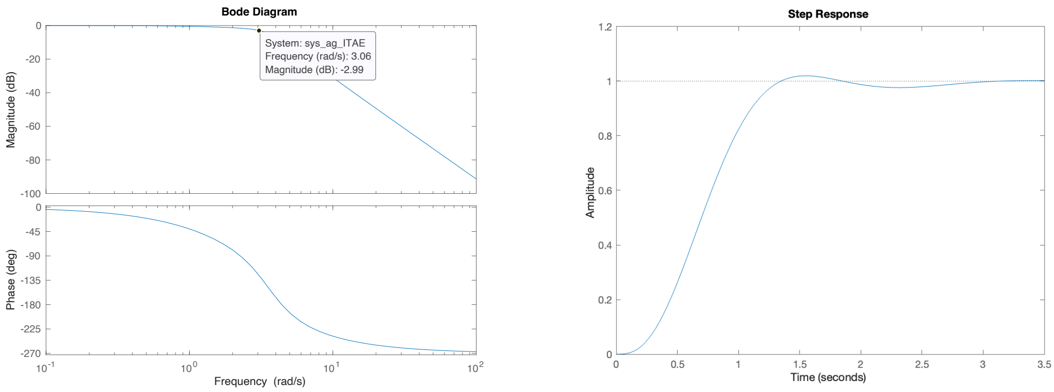

Figure 12 Additional system Augment state variable control results

Add disturbance

Add a small disturbance to these two systems, we can find the difference below.

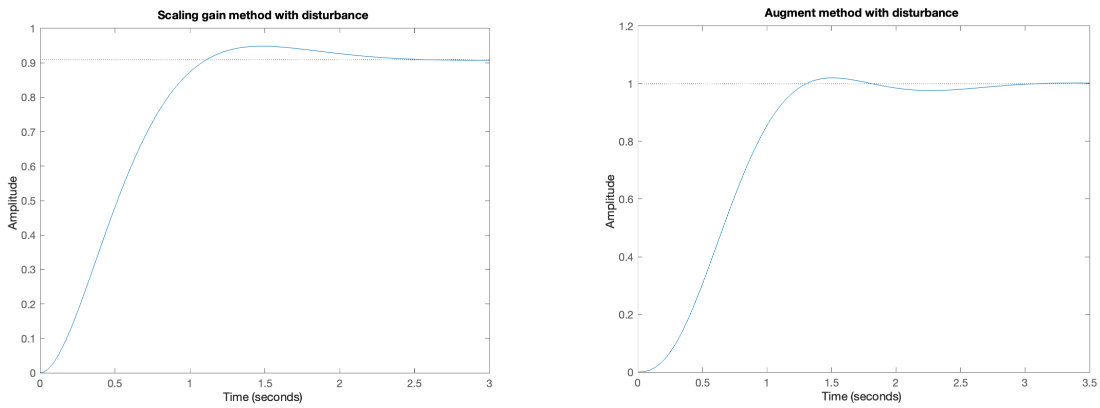

Figure 13 Disturbance analysis on additional system

In the absence of disturbances, the scaling gain method functions effectively. However, when disturbances are introduced, the outcomes are adversely affected, and the gain deviates from its nominal value of 1. Conversely, the augmented system exhibits superior disturbance rejection capabilities and robustness. Disturbances have minimal impact on the results, highlighting the augmented system's resilience in the face of external disturbances.

```
%% Additional exploration
% Open Loop System Analysis
F_Plant = [0, 1; -0.77, -3.51]; % Process Matrix
G_Plant = [0; 1.1]; % Input Matrix
H_Plant_State1 = [1,0]; % Output Matrix
% H_Plant_State2 = [0,1]; % Output Matrix Assuming x2 is the Output
sys1 = ss(F_Plant, G_Plant, H_Plant_State1, 0); % Use function 'ss' to establish state-space model
% sys2 = ss(F_Plant,G_Plant,H_Plant_State2,0);
figure(1) % open loop bode plot
bode(sys1) % Bode plot from u to x1; Use function 'bode' to plot bode graph
Poles = eig(F_Plant); % The open-loop poles are the eigenvalue of the original plant process matrix
% Use function 'eig' to calculate the eigenvalue of a square matrix
disp('The open-loop system transfer function poles are located at:')
display(Poles)
% controllable
ControllabilityMatrix = [G_Plant,F_Plant*G_Plant];
if (det(ControllabilityMatrix)==0)
```

```
    error('The original plant is not controllable.')
    % An 'error' command will stop the MATLAB program and pop-up an error message
    % If the controllability matrix has determinant of zero, the plant is not controllable.
else
    disp("This system is controllable")
end
%% Scaling gain
ITAEPoles = 3 * [-0.7071+0.7071*1i; -0.7071-0.7071*1i]; % ITAE Design Poles as a column vector
% In MATLAB, 'a+b*1i' is common way of defining complex value
K_ITAE = acker(F_Plant, G_Plant, ITAEPoles);
% Use function 'acker' to calculate the controller gain using Ackermann's Formula
sys_fb_ITAE = ss(F_Plant-G_Plant*K_ITAE, G_Plant, H_Plant_State1, 0);
Ks_2st = 1/dcgain(sys_fb_ITAE); % Calculate the steady state gain of the feedback system
% Use function 'dcgain' to calculate the steady state gain of a system
% The returned value of a function can be directly used as part of
% other calculations.
G_disturbance= [0; 1];
sys_fb_ITAE_dis = ss(F_Plant-G_Plant*K_ITAE, G_disturbance, H_Plant_State1, 0);
%% augment state variable
Fbar = [0,     1,   0;
        -0.77, -3.51, 0;
         1,     0,   0]; % The process matrix for the augmented system
Gu = [0; 1.1; 0];   % The input matrix for u
Gr = [0; 0; -1];     % The input matrix for r
 Gv = [0; -0.1; 0]; % Used for Disturbance Analysis; The input matrix for disturbance v
Hbar = [1, 0, 0];   % The output matrix for y

ITAEPoles = 3*[-0.7081; -0.5210+1.068*1i; -0.5210-1.068*1i];
K_ITAE = acker(Fbar,Gu,ITAEPoles);
sys_ag_ITAE = ss(Fbar-Gu*K_ITAE, Gr+Gv, Hbar, 0);
figure(12)
bode(sys_ag_ITAE)
figure(13)
step(sys_ag_ITAE)
grid on
```

Comparation

Simulation results indicate that the scaling gain approach exhibits a faster and more precise response, albeit limited to steady state. Its structure and calculations are simpler, enabling quick implementation. However, determining the scaling gain in practice may be complex, requiring numerous experiments. On the other hand, the augmented state variable approach performs exceptionally well in the presence of disturbances, excelling in dynamic performance. Nevertheless, it introduces complexity, potential for instability, and increased computational load. The structure of the augmented state variable approach is inherently more complex.

13. Multiple input multiple output system

In reality, the most of the systems have multiple inputs, so I want to know what will happen for MIMO system and do some simple research about it.
Based on the given system above, I directly add additional input on it and change the parameters of matrix G. Here is the new system.

$$\begin{bmatrix} \dot{x}_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1.2 & -3.71 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

The open loop response is

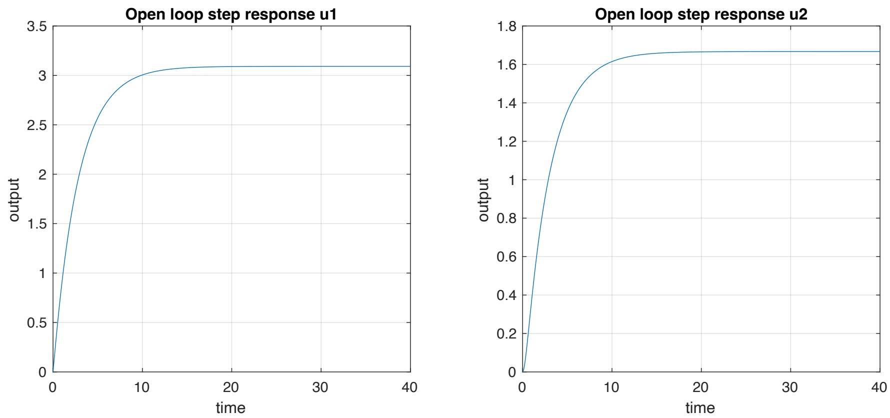

Figure 14 Open loop of MIMO system

To control the MIMO system, I found we have the similar method, In order to make the system stable or reach some characters we want, we can use full rank pole placement or unity rank method to place the poles.
Multiple input system can be transferred to single input form and solve using SISO method.
In this method we need choose q first, which is

$$u = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} v$$

v is scalar and the system is transferred to SISO system by using q.

$$\dot{x} = Ax + Bu = Ax + Bqu$$

Let

$$q = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$v = -k^T x$$

$$u = qv = -qk^T x = -Kx$$

Let

$$k = [k_1, k_2]$$

$$K = qk^T = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} [k_1, k_2]$$

The desired poles are -3 -3, the K is

0.1981	-0.2270
0.3963	-0.4541

The simulation results are:

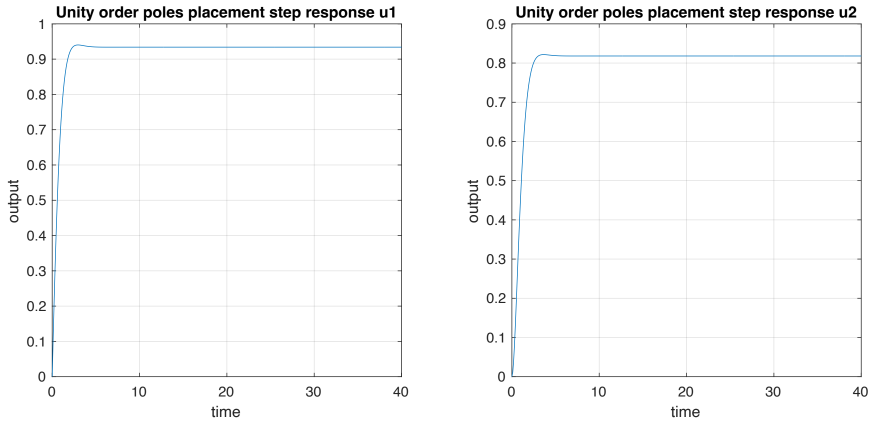

Figure 15 Unity order poles placement step response

With new poles the system response faster and only has a small overshoot. The under certain situation, this system performs much better. So we can see even for MIMO system, we still have many methods to control it.

We can also apply augmented matrix method on MIMO system and using states observer etc, but time is too limited I can spare too much time do it further.

```
clc
clear
close all
%% full order poles placement
A = [ 0 1; -1.2  -3.71];
B = [1 0; 0 2 ];
C = [1  0];
sys_open = ss(A, B, C, 0);

t=0:0.1:40;
len = size(t,2);
x0 = [0; 0];
u0=10*zeros(len,2);
u1=[ones(len,1),zeros(len,1)];
u2=[zeros(len,1),ones(len,1)];
% zero inputs and x0 initial state
[y, tout, x]=lsim(sys_open, u1, t, x0);
figure(1)
subplot(121)
plot(t, y)
grid on
%legend('y1','y2')
xlabel('time')
ylabel('output')
title('Open loop step response u1')
subplot(122)
[y, tout, x]=lsim(sys_open, u2, t, x0);
plot(t, y)
grid on
```

```
%legend('y1','y2')
xlabel('time')
ylabel('output')
title('Open loop step response u2')

%% Unity Rank Method
% ITAEPoles_narrow = 1.5*[-0.7081; -0.5210+1.068*1i; -0.5210-1.068*1i];
q = [1; 2];
C2 = [B*q, A*B*q];
rank(C2) % 4
syms k1 k2
k = [k1 k2];
K = q*k;
Feedback_m = A - B*K;
ploy_f = charpoly(Feedback_m);

equ1 = ploy_f(2) == 3 ;
equ2 = ploy_f(3) == 3 ;
equs=[equ1, equ2];
res = (solve(equs));
k1 = double(res.k1);
k2 =double(res.k2);

K2 = q*[k1 k2];
sys1 = ss(A, B, C, 0);
Af=A-B*K2;
sys_close=ss(Af, B, C, 0);  %Should be careful about the B here.
%bode(sys_close)
%step(sys_close)

t=0:0.1:40;
len = size(t,2);
x0 = [0; 0];
u0=10*zeros(len,2);
u1=[ones(len,1),zeros(len,1)];
u2=[zeros(len,1),ones(len,1)];
% zero inputs and x0 initial state
[y, tout, x]=lsim(sys_close, u1, t, x0);
figure(1)
subplot(121)
plot(t, y)
grid on
%legend('y1','y2')
xlabel('time')
ylabel('output')
title('Unity order poles placement step response u1')
subplot(122)
[y, tout, x]=lsim(sys_close, u2, t, x0);
plot(t, y)
grid on
%legend('y1','y2')
xlabel('time')
ylabel('output')
title('Unity order poles placement step response u2')
```

15. Conclusion

This study delved into the intricacies of scaling gain and augmented state variable approaches, elucidating their respective merits and drawbacks. The augmented state variable method emerged as a robust solution, adeptly fulfilling steady-state requirements without necessitating scaling gain adjustments. The inquiry underscored the critical importance of method selection tailored to specific system requisites, striking a nuanced balance between precision, adaptability, and robustness. While MIMO system control offers avenues for advanced methods like state observers, decoupling, and LQR, the time constraints necessitate their exploration in subsequent courses. In essence, this investigation not only refines theoretical concepts learned in class but also stands as a practical enhancement to understanding control strategies in real-world applications.